**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Методы обработки и анализа медицинских изображений»**

**Тема: СИНТЕЗ ПАНОРАМНЫХ ИЗОБРАЖЕНИЙ**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 8308 |  | Сергеев В.С. |
|  |  | Петров Г.А. |
|  |  | Лукашов Н.С. |
| Преподаватель |  | Поздеев А.А. |

Санкт-Петербург

2023

**Цель работы**

Изучение методов синтеза панорамы и их практическая реализация.

**Теоретические основания**

Общепринятый алгоритм для синтеза панорам состоит из следующих шагов.

1. Нахождение характерных точек (в англоязычной литературе используют термин «ключевая» точка «key point») на исходных изображениях в зоне перекрытия. Под характерной точкой понимают некоторый малый фрагмент изображения, в котором как значение яркостного градиента, так и производная (скорость изменения) градиента по направлению высоки. Как правило, характерных точками являются различные углы на изображениях (рис. 3.1).

2. Нахождение в зоне перекрытия одних и тех же особенностей на различных снимках (рисунок 3.2). То есть определение соответствующих друг другу пар характерных точек

Согласованные пары точек — это основа для синтеза панорамы. С помощью установленных пар на последующих этапах решают задачи калибровки, трансформации и объединения снимков. В случае синтеза панорамы из нескольких кадров (частей) при неизвестном заранее расположении каждого снимка, на данном этапе проводят процедуру регистрации – идентификации местоположения отдельных изображений на общей панораме и установление взаимных соответствий характерных точек.

3. В современных программных пакетах следующим шагом алгоритма сшивки является калибровка изображений. Эта процедура направлена на минимизацию искажений объектива, оптических дефектов, различий экспозиции. С помощью информации о согласованных парах характерных точек минимизируют влияние дисторсии (геометрических искажений) объектива на точность сшивки панорамы.

4. Ключевым этапом является процедура идентификации параметров уравнений трансформации изображений и последующее преобразование фрагментов с объединением в единую панораму. Данный этап требует задания вида трансформации, которое определяется типом создаваемой панорамы. Например, при отсутствии, или несущественности перспективных искажений у фрагментов имеет смысл использовать аффинное преобразование. Примером такого изображения может быть «сшивка» панорамы из отсканированных частей единого документа (географической карты, картины, и т.п.). В общем случае применяют перспективное преобразование, учитывающее все возможные искажения снимков.

5. Заключительным этапом является блэндинг. Это комплексная процедура, направленная на повышение визуального качества панорамы, включающая выравнивание яркости и цветовой палитры фрагментов, маскирование «швов», удаление «призраков» (движущихся объектов). Кроме того, в блэндинг включают и процедуру проецирования панорамы на заданную поверхность – сферическую, цилиндрическую, эквидистантную и пр.

**Задание**

Разработать программу, реализующую синтез панорамы из двух фрагментов с помощью библиотеки OpenCV.

Необходимо подобрать два фрагмента (зона перекрытия изображений должна занимать не менее 25% их площади, желателен поворот или перспективное преобразование одного изображения относительно другого) осуществить синтез панорамы.

При работе программы должны быть визуализированы основные этапы:

* + изображения с найденными ключевыми точками;
  + результат процедуры поиска согласованных пар;
  + результат трансформации изображения (один из снимков является опорным и остается без изменений, трансформируют только второй фрагмент);
  + результат построения панорамы.

**Ход работы**

Изображения с найденными ключевыми точками рисунках 1 и 2 соответственно.



Рисунок 1 – Левое изображение с ключевыми точками



Рисунок 2 – Правое изображение с ключевыми точками

Результаты процедуры поиска согласованных пар представлен на рисунке 3.



Рисунок 3 – Результат процедуры поиска согласованных пар

Результат трансформации правого изображения представлен на рисунке 4.



Рисунок 4 – Результат трансформации правого изображения

Результат построения панорамы представлен на рисунке 5.



Рисунок 5 – Полученная панорама

**Вывод**

В ходе выполнения лабораторной работы были изучены методы синтеза панорамы, а также реализованы на практике с визуализацией результатов на каждом шаге общего алгоритма.

Приложение А. Листинг исходного кода.

#include <iostream>

#include <opencv2\opencv.hpp>

#include <algorithm>

**using** **namespace** std;

**using** **namespace** cv;

**void** **erosion**(**const** Mat& input\_img, Mat& output\_img, **int** n = **1**)

{

output\_img = Mat::zeros(input\_img.size(), CV\_8U);

**for** (**int** i = n; i < input\_img.cols - n; i++)

**for** (**int** j = n; j < input\_img.rows - n; j++) {

uchar pix\_value = input\_img.at<uchar>(j, i);

**float** min = **255**;

**for** (**int** ii = -n; ii <= n; ii++)

**for** (**int** jj = -n; jj <= n; jj++) {

uchar Y = input\_img.at<uchar>(j + jj, i + ii);

**if** (Y < min)

min = Y;

}

output\_img.at<uchar>(j, i) = min;

}

}

**void** **dilation**(**const** Mat& input\_img, Mat& output\_img, **int** n = **1**)

{

output\_img = Mat::zeros(input\_img.size(), CV\_8U);

**for** (**int** i = n; i < input\_img.cols - n; i++)

**for** (**int** j = n; j < input\_img.rows - n; j++) {

uchar pix\_value = input\_img.at<uchar>(j, i);

**float** max = **0**;

**for** (**int** ii = -n; ii <= n; ii++)

**for** (**int** jj = -n; jj <= n; jj++) {

uchar Y = input\_img.at<uchar>(j + jj, i + ii);

**if** (Y > max)

max = Y;

}

output\_img.at<uchar>(j, i) = max;

}

}

**void** **opening**(**const** Mat& input\_img, Mat& output\_img, **int** n = **1**)

{

Mat buf;

erosion(input\_img, buf, n);

dilation(buf, output\_img, n);

}

**void** **closing**(**const** Mat& input\_img, Mat& output\_img, **int** n = **1**)

{

Mat buf;

dilation(input\_img, buf, n);

erosion(buf, output\_img, n);

}

**void** **contour**(**const** Mat& input\_img, Mat& output\_img, **int** n = **1**)

{

Mat buf1, buf2;

dilation(input\_img, buf1, n);

erosion(input\_img, buf2, n);

output\_img = buf1 - buf2;

//erosion(input\_img, buf2, n);

//output\_img = input\_img - buf2;

}

**void** **MG**(**const** Mat& input\_img, Mat& output\_img)

{

output\_img = Mat::zeros(input\_img.size(), CV\_8U);

Mat bufs[**3**];

**for** (**int** i = **0**; i < **3**; i += **1**) {

Mat buf1, buf2;

dilation(input\_img, buf1, i + **1**);

erosion(input\_img, buf2, i + **1**);

erosion(buf1 - buf2, bufs[i], i);

}

**for** (**int** i = **0**; i < input\_img.cols; i += **1**)

**for** (**int** j = **0**; j < input\_img.rows; j += **1**) {

**float** result = **0**;

**for** (**int** q = **0**; q < **3**; q += **1**) {

result += bufs[q].at<uchar>(j, i);

}

output\_img.at<uchar>(j, i) = (uchar)(result / **3**);

}

}

**int** **main**()

{

string name = "";

**int** colorType = **0**;

cout << "File name: ";

cin >> name;

Mat gray\_img = imread(name, colorType);

**if** (gray\_img.empty()) {

**return** **0**;

}

Mat erosion\_img, dilation\_img, open\_img, close\_img, contour\_img, MG\_img;

erosion(gray\_img, erosion\_img);

dilation(gray\_img, dilation\_img);

opening(gray\_img, open\_img);

closing(gray\_img, close\_img);

contour(gray\_img, contour\_img);

MG(gray\_img, MG\_img);

Mat bin\_img, bin\_erosion\_img, bin\_dilation\_img, bin\_open\_img, bin\_close\_img, bin\_contour\_img, bin\_MG\_img;

threshold(gray\_img, bin\_img, **150**, **255**, THRESH\_BINARY); //THRESH\_BINARY\_INV

erosion(bin\_img, bin\_erosion\_img);

dilation(bin\_img, bin\_dilation\_img);

opening(bin\_img, bin\_open\_img);

closing(bin\_img, bin\_close\_img);

contour(bin\_img, bin\_contour\_img);

MG(bin\_img, bin\_MG\_img);

namedWindow("gray\_img", WINDOW\_AUTOSIZE);

imshow("gray\_img", gray\_img);

namedWindow("erosion\_img", WINDOW\_AUTOSIZE);

imshow("erosion\_img", erosion\_img);

namedWindow("dilation\_img", WINDOW\_AUTOSIZE);

imshow("dilation\_img", dilation\_img);

namedWindow("open\_img", WINDOW\_AUTOSIZE);

imshow("open\_img", open\_img);

namedWindow("close\_img", WINDOW\_AUTOSIZE);

imshow("close\_img", close\_img);

namedWindow("contour\_img", WINDOW\_AUTOSIZE);

imshow("contour\_img", contour\_img);

namedWindow("MG\_img", WINDOW\_AUTOSIZE);

imshow("MG\_img", MG\_img);

namedWindow("bin\_img", WINDOW\_AUTOSIZE);

imshow("bin\_img", bin\_img);

namedWindow("bin\_erosion\_img", WINDOW\_AUTOSIZE);

imshow("bin\_erosion\_img", bin\_erosion\_img);

namedWindow("bin\_dilation\_img", WINDOW\_AUTOSIZE);

imshow("bin\_dilation\_img", bin\_dilation\_img);

namedWindow("bin\_open\_img", WINDOW\_AUTOSIZE);

imshow("bin\_open\_img", bin\_open\_img);

namedWindow("bin\_close\_img", WINDOW\_AUTOSIZE);

imshow("bin\_close\_img", bin\_close\_img);

waitKey(**0**);

destroyWindow("gray\_img");

destroyWindow("erosion\_img");

destroyWindow("dilation\_img");

destroyWindow("open\_img");

destroyWindow("close\_img");

destroyWindow("contour\_img");

destroyWindow("MG\_img");

destroyWindow("bin\_img");

destroyWindow("bin\_erosion\_img");

destroyWindow("bin\_dilation\_img");

destroyWindow("bin\_open\_img");

destroyWindow("bin\_close\_img");

**return** **0**;

}